NAG Library Function Document

nag_summary_stats_freq (g01adc)

1 Purpose

nag_summary_stats_freq (g01adc) calculates the mean, standard deviation and coefficients of skewness and kurtosis for data grouped in a frequency distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_summary_stats_freq (Integer k, const double x[],
     const Integer ifreq[], double *xmean, double *xsd, double *xskew,
     double *xkurt, Integer *n, NagError *fail)
```

3 Description

The input data consist of a univariate frequency distribution, denoted by f_i, for $i = 1, 2, \ldots, k-1$, and the boundary values of the classes x_i, for $i = 1, 2, \ldots, k$. Thus the frequency associated with the interval (x_i, x_{i+1}) is f_i, and nag_summary_stats_freq (g01adc) assumes that all the values in this interval are concentrated at the point

$$y_i = (x_{i+1} + x_i)/2, \quad i = 1, 2, \ldots, k-1.$$

The following quantities are calculated:

(a) total frequency,

$$n = \sum_{i=1}^{k-1} f_i.$$

(b) mean,

$$\bar{y} = \frac{\sum_{i=1}^{k-1} f_i y_i}{n}.$$

(c) standard deviation,

$$s_2 = \sqrt{\frac{\sum_{i=1}^{k-1} f_i (y_i - \bar{y})^2}{(n-1)}}, \quad n \geq 2.$$

(d) coefficient of skewness,

$$s_3 = \frac{\sum_{i=1}^{k-1} f_i (y_i - \bar{y})^3}{(n-1) \times s_2^3}, \quad n \geq 2.$$

(e) coefficient of kurtosis,

$$s_4 = \frac{\sum_{i=1}^{k-1} f_i (y_i - \bar{y})^4}{(n-1) \times s_2^4} - 3, \quad n \geq 2.$$

The function has been developed primarily for groupings of a continuous variable. If, however, the function is to be used on the frequency distribution of a discrete variable, taking the values $y_1, \ldots, y_{k-1}$, then the boundary values for the classes may be defined as follows:

(i) for $k > 2$,

$$\begin{aligned} x_1 &= (3y_1 - y_2)/2 \\ x_j &= \big(y_{j-1} + y_j\big)/2, \qquad j = 2, \ldots, k-1 \\ x_k &= (3y_{k-1} - y_{k-2})/2 \end{aligned}$$

(ii) for $k = 2$,

$$x_1 = y_1 - a \quad \text{and} \quad x_2 = y_1 + a \quad \text{for any } a > 0.$$

4 References

None.

5 Arguments

1: **k** – Integer *Input*

On entry: k, the number of class boundaries, which is one more than the number of classes of the frequency distribution.

Constraint: $\mathbf{k} > 1$.

2: **x**[**k**] – const double *Input*

On entry: the elements of **x** must contain the boundary values of the classes in ascending order, so that class i is bounded by the values in $\mathbf{x}[i-1]$ and $\mathbf{x}[i]$, for $i = 1, 2, \ldots, k-1$.

Constraint: $\mathbf{x}[i] < \mathbf{x}[i+1]$, for $i = 0, 1, \ldots, k-2$.

3: **ifreq**[**k**] – const Integer *Input*

On entry: the ith element of **ifreq** must contain the frequency associated with the ith class, for $i = 1, 2, \ldots, k-1$. $\mathbf{ifreq}[k-1]$ is not used by the function.

Constraints:

$$\mathbf{ifreq}[i-1] \geq 0, \text{ for } i = 1, 2, \ldots, k-1;$$
$$\sum_{i=1}^{k-1} \mathbf{ifreq}[i-1] > 0.$$

4: **xmean** – double * *Output*

On exit: the mean value, $\bar{y}$.

5: **xsd** – double * *Output*

On exit: the standard deviation, s_2.

6: **xskew** – double * *Output*

On exit: the coefficient of skewness, s_3.

7: **xkurt** – double * *Output*

On exit: the coefficient of kurtosis, s_4.

8: **n** – Integer * *Output*

On exit: the total frequency, n.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_FREQ_CONS

Either **ifreq**$[i] < 0$ for some i, or the sum of frequencies is zero.

NE_FREQ_SUM

The total frequency is less than 2.

NE_INT

On entry, $\mathbf{k} = \langle value \rangle$.
Constraint: $\mathbf{k} > 1$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_NOT_INCREASING

On entry, $I = \langle value \rangle$, $\mathbf{x}[I-2] = \langle value \rangle$ and $\mathbf{x}[I-1] = \langle value \rangle$.
Constraint: $\mathbf{x}[I-2] \leq \mathbf{x}[I-1]$.

7 Accuracy

The method used is believed to be stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_summary_stats_freq (g01adc) increases linearly with k.

10 Example

In the example program, NPROB determines the number of sets of data to be analysed. For each analysis, the boundary values of the classes and the frequencies are read. After nag_summary_stats_freq (g01adc) has been successfully called, the input data and calculated quantities are printed. In the example, there is one set of data, with 14 classes.

10.1 Program Text

```
/* nag_summary_stats_freq (g01adc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{

  /* Scalars */
  double   xsd, xskew, xkurt, xmean;
  Integer  exit_status = 0, i, j, k, kmin1, n, nprob;

  NagError fail;

  /* Arrays */
  double   *x = 0;
  Integer  *ifreq = 0;

  INIT_FAIL(fail);

  printf("nag_summary_stats_freq (g01adc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &nprob);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &nprob);
#endif
  for (j = 1; j <= nprob; ++j)
    {
#ifdef _WIN32
      scanf_s("%"NAG_IFMT"%*[^\n] ", &kmin1);
#else
      scanf("%"NAG_IFMT"%*[^\n] ", &kmin1);
#endif
      k = kmin1 + 1;

      /* Allocate memory */
      if (!(x = NAG_ALLOC(k, double)) ||
          !(ifreq = NAG_ALLOC(k, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      for (i = 1; i <= kmin1; ++i)
```

```
#ifdef _WIN32
        scanf_s("%lf%"NAG_IFMT"", &x[i - 1], &ifreq[i - 1]);
#else
        scanf("%lf%"NAG_IFMT"", &x[i - 1], &ifreq[i - 1]);
#endif
#ifdef _WIN32
      scanf_s("%lf%*[^\n] ", &x[k - 1]);
#else
      scanf("%lf%*[^\n] ", &x[k - 1]);
#endif

      printf("\nProblem %4"NAG_IFMT"\n", j);
      printf("Number of classes %4"NAG_IFMT"\n", kmin1);

      /* nag_summary_stats_freq (g01adc).
       * Mean, variance, skewness, kurtosis, etc., one variable,
       * from frequency table
       */
      nag_summary_stats_freq(k, x, ifreq, &xmean, &xsd, &xskew, &xkurt, &n,
                             &fail);

      if (fail.code == NE_NOERROR)
        {
          printf("Successful call of "
                  "nag_summary_stats_freq (g01adc)\n\n");
          printf("          Class          Frequency\n\n");
          for (i = 1; i <= kmin1; ++i)
            printf("%10.2f%10.2f%12"NAG_IFMT"\n", x[i-1], x[i],
                    ifreq[i-1]);

          printf("\n Mean %16.4f\n", xmean);
          printf(" Std devn%13.4f\n", xsd);
          printf(" Skewness%13.4f\n", xskew);
          printf(" Kurtosis%13.4f\n", xkurt);
          printf(" Number of cases%8"NAG_IFMT"\n", n);
        }
      else
        {
          printf("Error from nag_summary_stats_freq (g01adc).\n%s\n",
                  fail.message);
          exit_status = 1;
        }
      NAG_FREE(x);
      NAG_FREE(ifreq);
    }
 END:
  NAG_FREE(x);
  NAG_FREE(ifreq);
  return exit_status;
}
```

10.2 Program Data

```
nag_summary_stats_freq (g01adc) Example Program Data
 1
14
     9.3       3      12      19      14      52      16      96
      18     121      20     115      22      86      24      70
      26      49      28      31      30      16      32       6
      34       8      36       7    39.7
```

10.3 Program Results

```
nag_summary_stats_freq (g01adc) Example Program Results

Problem    1
Number of classes   14
Successful call of nag_summary_stats_freq (g01adc)

          Class          Frequency
```

```
      9.30     12.00           3
     12.00     14.00          19
     14.00     16.00          52
     16.00     18.00          96
     18.00     20.00         121
     20.00     22.00         115
     22.00     24.00          86
     24.00     26.00          70
     26.00     28.00          49
     28.00     30.00          31
     30.00     32.00          16
     32.00     34.00           6
     34.00     36.00           8
     36.00     39.70           7

 Mean          21.4932
 Std devn       4.9325
 Skewness       0.7072
 Kurtosis       0.5738
 Number of cases     679
```